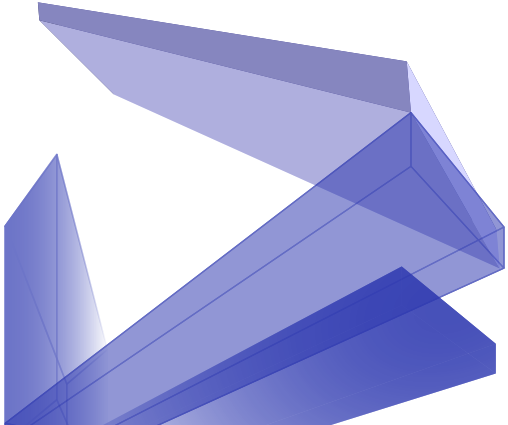


grain

D Language Library for Deep Learning

ML Meetup KANSAI #3

4. Oct. 2018

D Language for Deep Learning

great language

- **like C++**: fast, strongly typed, LLVM/GCC backend
- **like Python**: simple, lightweight, jupyter support[1]

great libraries

- **mir**: N-D slice algorithm[2], zero-cost numpy conversion[3]
- **dcompute**: CUDA kernel DSL[4]

---

[1] `https://github.com/kaleidicassociates/jupyterd`
[2] `https://github.com/libmir/mir-algorithm`
[3] `https://github.com/ShigekiKarita/mir-pybuffer`
[4] `https://github.com/libmir/dcompute`

grain

deep learning framework for D language

- https://github.com/ShigekiKarita/grain
- boost software license 1.0

philosophy

- **DYNAMIC**: like chainer and pytorch
- **SAFE**: statically typed variable and function
- **LIGHT**: simple like Python, small like C++
- **FAST**: mir and CUDA backend

grain is dynamic

like chainer ...

```
foreach (epoch; 0 .. 10) {
  foreach (i; niter.permutation) {
    auto xs = inputs[i].variable;
    auto ts = targets[i].variable;
    auto ys = model(xs);
    auto loss = crossEntropy(ys, ts);
    auto acc = accuracy(ys, ts);
    model.zeroGrad();
    loss.backward();
    optimizer.update();
  }
}
```

grain is dynamic

like chainer but statically typed and optimized.

```
foreach (epoch; 0 .. 10) {
  foreach (i; niter.permutation) {
    Variable!(float, 3, HostStorage) xs = inputs[i].variable;
    Variable!(int, 1, HostStorage) ts = targets[i].variable;
    Variable!(float, 2, HostStorage) ys = model(xs);
    Variable!(float, 0, HostStorage)loss =crossEntropy(ys, ts);
    float acc = accuracy(ys, ts);
    model.zeroGrad();
    loss.backward();
    optimizer.update();
  }
}
```

grain is safe

function is also statically typed and optimized.

```
struct Sigmoid(T, size_t dim) {
  Variable!(T, dim, HostStorage) y;
  // D compiler guarantees that nothing throws exception
  nothrow forward(Variable!(T, dim, HostStorage) x) {
    auto y = x.sliced.map!(a => tanh(a * 0.5) * 0.5 + 0.5)
              .slice.variable(x.requiresGrad);
    if (x.requiresGrad) this.y = y;
    return y;
  }
  nothrow backward(Variable!(T, dim, HostStorage) gy) {
    auto ys = this.y.sliced;
    return slice((1.0 - ys) * ys * gy.sliced).variable;
  }
  mixin FunctionCommon; // inject type checking
}
```

grain is safe

Chainer/PyTorch issues

- runtime overhead
  - for-loop, dynamic dispatch, func call
- runtime error:
  - type error, exception safety, memory leak

D solved these issues by compiler techniques and GC

grain is a lightweight language

Jupyter notebook support [5]

jupyter tutorial (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted

Run Code

In [1]:
```
// numpy-like broadcasting is available
static assert(!__traits(compiles, [[0.1f], [0.3f]].variable + [0.2f].variable));
// but you need to make them same dimensions
[[0.1f, 0.2f], [0.3f, 0.4f]].variable + [[0.2f]].variable
```

Out[1]:
```
Variable!(float, dim=2, HostStorage(T))(data=[0.3, 0.4, 0.5, 0.6], shape=[2, 2], strid
es=[2, 1])
```

Defining new autograd functions

Under the hood, each primitive autograd operator is really two functions that operate on Variables. The forward function computes output Variables from input Variable. The backward

[5] https://github.com/ShigekiKarita/grain/blob/master/tutorial.ipynb

grain is a lightweight library

smaller footprint library

framework	type	lines	mb
grain	**static lib**	**12,431**	**0.6**
chainer	python code	162,106	6
pytorch	dynamic lib	193,754	911
tensorflow	dynamic lib	130,475	285

smaller exe file (MNIST : 1.8MB, CIFAR: 2.3MB)

grain is fast

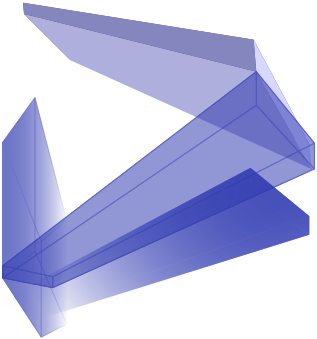
task	backend	framework	train (mb/s)	test (mb/s)
mnist	CUDA	grain	**300**	-
		chainer	-	-
		pytorch	200	-
	CPU	grain	**90**	-
		chainer	-	-
		pytorch	80	-
ptb	CUDA	grain	-	-
		pytorch	-	-

- grain 0.0.10, chainer 4.4.0, pytorch 0.4.1, CUDA9.2, CUDNN7

summary

grain: deep learning framework for D language

- **DYNAMIC**: like chainer and pytorch
- **SAFE**: statically typed variable and function
- **LIGHT**: simple like Python, small like C++
- **FAST**: mir and CUDA backend



CONTRIBUTION IS WELCOME!

https://github.com/ShigekiKarita/grain

examples

- Image recognition (mnist, cifar)
- Language modeling (shakespere, ptb)
- WIP
  - Reinforcement learning (cartpole)
  - Speech recognition (librispeech)
  - Machine translation (anki)

future work

- probabilistic programming
- lazy evaluation mode
- low resource environment (RasberryPi)